Information Systems Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Deep Learning of Spatiotemporal Patterns for Urban Mobility Prediction Using Big Data

Yun Wang, Faiz Currim, Sudha Ram

To cite this article:
Yun Wang, Faiz Currim, Sudha Ram (2022) Deep Learning of Spatiotemporal Patterns for Urban Mobility Prediction Using Big Data. Information Systems Research 33(2):579-598. https://doi.org/10.1287/isre.2021.1072

http://pubsonline.informs.org/journal/isre

INFORMATION SYSTEMS RESEARCH
Vol. 33, No. 2, June 2022, pp. 579–598
ISSN 1047-7047 (print), ISSN 1526-5536 (online)

Deep Learning of Spatiotemporal Patterns for Urban Mobility Prediction Using Big Data

Yun Wang,[a] Faiz Currim,[b,*] Sudha Ram[b]

[a] Microsoft; [b] Department of Management Information Systems, Eller College of Management, University of Arizona, Tucson, Arizona 85721
*Corresponding author

Contact: yunw@email.arizona.edu (YW); currim@arizona.edu, https://orcid.org/0000-0002-5025-811X (FC); ram@arizona.edu, https://orcid.org/0000-0001-6053-1311 (SR)

Received: August 9, 2017
Revised: May 30, 2018; August 3, 2019; August 11, 2021
Accepted: August 24, 2021
Published Online in Articles in Advance: February 11, 2022

https://doi.org/10.1287/isre.2021.1072

Abstract. Timely and accurate prediction of human movement in urban areas offers instructive insights into transportation management, public safety, and location-based services, to name a few. Yet, modeling urban mobility is challenging and complex because of the spatiotemporal dynamics of movement behavior and the influence of exogenous factors such as weather, holidays, and local events. In this paper, we use bus transportation as a proxy to mine spatiotemporal travel patterns. We propose a deep-learning-based urban mobility prediction model that collectively forecasts passenger flows between pairs of city regions in an origin-destination (OD) matrix. We first process OD matrices in a convolutional neural network to capture spatial correlations. Intermediate results are reconstructed into three multivariate time series: hourly, daily, and weekly time series. Each time series is aggregated in a long short-term memory (LSTM) network with a novel attention mechanism to guide the aggregation. In addition, our model is context-aware by using contextual embeddings learned from exogenous factors. We dynamically merge results from LSTM components and context embeddings in a late fusion network to make a final prediction. The proposed model is implemented and evaluated using a large-scale transportation data set of more than 200 million bus trips with a suite of Big Data technologies developed for data processing. Through performance comparison, we show that our approach achieves sizable accuracy improvements in urban mobility prediction. Our work has major implications for efficient transportation system design and performance improvement. The proposed deep neural network structure is generally applicable for sequential graph data prediction.

History: Olivia Sheng, Senior Editor; Xue Bai, Associate Editor.
Funding: This work was funded in part by a grant from the Edson Queiroz Foundation, Brazil.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/isre.2021.1072.

Keywords: big data • deep learning • smart transportation • predictive modeling

1. Introduction

With the global trend toward urbanization, the concept of "smart cities," as a data and information systems–supported vision of urban development, has gained increasing attention within both academia and industry. Large-scale human-sensing data sets facilitated by wireless technologies, such as global positioning system (GPS), radio-frequency identification (RFID) chips, and contactless smart cards have become basic ingredients for the new wave of urban development. Indeed, the availability of massive real-time location data opens the opportunity to revolutionize human mobility modeling. Understanding human mobility offers instructive insights into various aspects of urban management (Yang et al. 2016). In particular, timely and accurate prediction of human movement in urban areas is of great importance for adaptive and demand-driven service management that characterize the term "smart" (in smart cities).

Despite the rich and varied amount of data, modeling urban mobility is still challenging because of three major factors that affect travel behaviors. First, human convergence and dispersion exhibit mixed temporal dependencies on different time scales (González et al. 2008). For instance, on a typical weekday, crowding on public transport reaches its peak during rush hours and drops to a low in the early morning or late at night. On a weekly basis, travel behaviors are similar among adjacent weekdays but change tremendously when it comes to weekends. Second, urban dynamics depend on the spatial distribution of regions related to diverse city functions such as providing space for residence, recreation, industry, and transit transfer facilities. Literature has shown that periodic and bursty

patterns of public transportation ridership vary across the transit network (Sun et al. 2014). Third, urban mobility is influenced by exogenous factors such as the kind of holiday, weather, special events—each of which can also interact with spatiotemporal factors (Strover and Edward 2012). The factor impacts vary by transport mode, time, and season.

To analyze this problem, we use data from a public transportation network in a large city of a developing country as a proxy to characterize and model urban mobility. In this paper, we address the research problem of predicting short-term bus passenger ridership between urban regions, which are identified as sets of bus stops with similar accessibility and travel attractions, using deep learning. The motivation for our research focus is driven by the projection that much of the urban sprawl is expected to occur in cities of developing nations where public transportation is a backbone of urban development (Gwilliam 2003). Public transportation in populous cities is often plagued by passenger overcrowding (Farkas et al. 2015). Short-term forecasting ridership demand on a granular level provides urban planners with real-time evidence to support dynamic service scheduling for better operational efficiency and lower management cost.

Traditional approaches for ridership forecasting focus on the prediction on a small section of transit network and assume the demand is a linear combination of human engineered factors, which is too simplistic to capture complex relationships among factors affecting passenger flows. On the other hand, deep learning methods do not require extensive human effort and have shown remarkable performance with multivariate time series data that exhibit nonlinear properties. The advantages of modeling human mobility using deep neural networks are twofold. First, recent developments in deep learning have shown promise for self-taught learning of spatiotemporal features. Particular neural networks such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and network embedding (NE) are efficient in encoding representations from spatial dependencies, temporal sequences, and exogenous factors. Second, learned representations can be stacked strategically to form a deeper network where nonlinear interactions between all factors and their impacts on human movement can be learned in an end-to-end manner. In light of these benefits, we present a novel deep neural network architecture to collectively predict short-term passenger flow between each pair of urban regions.

We model citywide bus trips in a directed graph where vertices are urban regions, and directed edges indicate aggregated passenger volume. We update the graph periodically and represent it in an origin-destination (OD) matrix. The proposed model is constructed by stacking three neural network components. The first component is a CNN that processes OD matrices to extract spatial dependencies. The receptive fields of the convolutional kernels are specially designed to capture the mass gatherings in bus travels. In the second component, CNN outputs are organized into three multivariate time series of different intervals, namely hourly, daily, and weekly time series. We used RNN to learn patterns from sequential matrices (Lukoševičius and Jaeger 2009). Of particular interest is the long short-term memory (LSTM) that is capable of encoding long-term dependencies (Hochreiter and Schmidhuber 1997). Each time series is aggregated in an LSTM network to extract inherent and nonlinear representation for prediction. Furthermore, because exogenous factors are not explicitly encoded in OD matrices, we attach a neural attention mechanism on top of LSTM networks to guide the aggregation. To effectively calculate neural attentions, we convert exogenous factors into real-valued representations by considering their effects on ridership demand. We call these vectors context embedding because they make the prediction context aware. Last, we introduce a late fusion network as the third component in the framework, which dynamically merges predictions from LSTM components and adjustments from context embedding to produce a final prediction. The entire deep neural network is trained in an end-to-end manner using bus GPS trajectory and automatic fare collection (AFC) data on a system-wide scale. To facilitate the training, we invented a suite of big data techniques to process the raw data, including a distributed algorithm that calculates passenger movement and bus travel time from sensor streams. These techniques provide a relatively economical solution (compared with systems that use expensive passenger monitoring equipment on each bus, for example) for monitoring public transportation in cities where expensive digital infrastructure is commonly constrained by restricted resources.

Our contributions of the paper can be summarized as follows:

i. We formulate complex ridership patterns as time-varying mobility graphs (OD matrix representation). The graphs are generalizable to multiple levels of spatial granularities (e.g., regions or routes) and different mobility patterns (e.g., buses, metros, and taxis) on a large scale.

ii. We incorporate complex and nonlinear spatial, temporal, and contextual features into the ridership forecasting problem through automatic feature engineering (and creative adaptation of techniques such as contextual embedding).

iii. We develop a hybrid deep learning architecture that incorporates and adapts various types of deep learning techniques (CNN, LSTM, Feed-forward Neural Network (FNN)), and domain-specific feature attention and fusion mechanisms to simultaneously optimize feature extraction and prediction.

The remainder of this paper is organized as follows. Section 2 provides a condensed review of previous work, identifies the research gaps, and discusses deep learning literature that motivates our design. Section 3 formulates the problem and provides an overview of the design of our approach. Section 4 presents architectural details of each neural network component in the predictive model. Section 5 draws on a case of use in one of the largest cities in Brazil to validate the proposed approach in a real-world setting. Finally, conclusions are drawn with future study directions in Section 6.

2. Related Work

In this section, we discuss briefly recent research related to ridership forecasting and their limitations, followed by an introduction to the deep learning techniques that lead to the design of this work.

2.1. Ridership Forecasting in Transportation Domain

Ridership forecasting is one of the fundamental functions that feature intelligent transportation management. Broadly speaking, ridership forecasting can be considered as a time series forecasting problem. Many statistical models have been be applied to solve this problem including autoregressive (AR), autoregressive integrated moving average (ARIMA), and seasonal ARIMA (SARIMA) models. A sequential framework for short-term bus passenger flow prediction has been proposed based on SARIMA (Gong et al. 2014). To make prediction more adaptive, a hybrid model is discussed in which different ARIMA-family models are mixed to capture different features of time series (Xue et al. 2015). However, given the variation of these time series models, it is still difficult to say that which one of these methods is superior to others for ridership demand forecasting. One of the reasons for this is that these models were developed with data from a small part of the transit network and ignore the important spatiotemporal features on a large scale, which limit their generalizability.

Another limitation of ARIMA-family models is that they solely investigate statistical patterns exhibited by time series data. Because of the nonlinear nature of time series, researchers have used machine learning methods where information from other sources can also be included in the learning. For example, a neural network–based forecasting examined the influence of holidays and drastic changes in weather patterns (Peng et al. 1992). Other commonly used machine learning methods include support vector regression (SVR), Gaussian process, and Bayesian network (Sun et al. 2006). Machine learning methods for time series forecasting often require developing hand-crafted features (Längkvist et al. 2014). However, feature engineering can be time consuming, task specific, and sometimes only feasible using domain knowledge.

From a data representation perspective, many of the ridership forecasting studies have been carried out with the OD matrix. Urban transit networks can naturally form an OD matrix, making this representation an ideal source to model urban mobility for better operational efficiency. Nevertheless, transportation data can suffer from the problem of data incompleteness. For instance, passenger alighting locations are often missed in AFC records because passenger cards may only be swiped when boarding. Several studies attempt to address this by reconstructing OD chains using either easy-to-implement rule-based approaches (Wang et al. 2011) or more advanced generative probabilistic models (Li and Cassidy 2007, Munizaga and Palma 2012, Yuan et al. 2013). Notwithstanding the effort needed for data preprocessing, transportation data are gaining more attention owing to increased research in smart cities (Wang et al. 2016, Ullah et al. 2020).

2.2. Gap Analysis

In the transportation domain, deep learning technology has received increasing attention (Lv et al. 2015, Hasija et al. 2020). Related to urban mobility prediction, multiple authors have focused on predicting passenger flows for a single station or line (Du et al. 2020, Wu et al. 2021) including using hybrid approaches (Liu and Chen 2017, Yang et al. 2021). In terms of regional movements, a deep residual learning approach for crowd flows prediction has been proposed (Zhang et al. 2017), as well as a hybrid approach that uses CNN and LSTM for predicting taxi demand (Yao et al. 2018). Although both of the latter papers focus on crowd flows between adjacent neighborhoods, our work predicts ridership between two arbitrary areas. The proposed model trained on ridership forecasting can serve as transfer learner that encodes urban mobility (in the form of OD matrices) to be fed into subsequent neuron layers and fine-tuned for other tasks in transportation domain. In reviewing previous attempts in ridership forecasting, we identified the following areas that are critical to the increase in performance and generalizability yet have not been fully explored.

2.2.1. Generic OD Matrix. Unlike previous works that build OD matrix from partial transit networks, constructing an OD matrix on an urban scale can substantially increase its generalizability. Using components of an urban transit system, such as bus stops or metro

stations, is the most straightforward way to build such an OD matrix. However, doing so may suffer from the design limitations of the specific system. For example, other than hub stations, most bus stops only have a handful of connections, causing performance issues related using sparse matrices. A general approach that takes into consideration travel attractions and accessibility is more desirable, so that ridership in multimodal transit systems can be aggregated in a homogenous representation.

2.2.2. Heterogeneous Feature Fusion. As previously mentioned, obstacles to forecasting ridership is not only the modeling of the time series but also the more complex correlations among spatial, temporal, and exogenous factors. Therefore, we need to develop a statistical framework that can build links across heterogeneous features for a more comprehensive coverage. In particular, the framework should be effective at learning from categorical features by encoding them into a suitable feature vectors.

2.2.3. End-to-End Learning. Traditional knowledge driven approaches have separate optimizations for feature extraction and ridership forecasting. The former often requires specific domain knowledge to be fine-tuned. In this work, we argue that a pipeline architecture that optimizes both feature extraction and ridership forecasting under a set of criteria (better forecasting accuracy) is more suitable in today's big data era. An end-to-end pipeline is more generalizable as it does not require excessive prior knowledge. Ideally, it can gain knowledge (meaningful self-taught features) by itself in a data-driven manner.

2.3. Deep Learning Methods

In this section, we summarize deep learning methods used in this paper. Our approach uses multiple techniques (CNN, LSTM, context embedding, attention, and fusion) to bridge the gap seen in the previous section. To evaluate our work, we also compare our results with the performance of TDNN and CRBM.

2.3.1. CNN. CNNs are a kind of FNN that use convolutions instead of matrix multiplication in one of their layers (Goodfellow et al. 2016). CNNs are essentially layers of convolutions with nonlinear activation functions and optional down sampling (pooling). A combination of these operations learns local and global correlations in high-dimensional inputs. Among deep learning architectures, CNNs have become a dominant one in computer vision for their ability to explore the spatial correlations in computer vision and pattern recognition tasks (Krizhevsky et al. 2012).

2.3.2. RNN/LSTM. RNNs have self-connecting loops to maintain information from previous passes. Hidden state neurons in RNNs are used as a memory to generalize patterns across sequences of inputs. From a modeling perspective, we argue that an RNN-based neural network structure is more appropriate to model temporal dependence in time series. LSTM is an extension of RNN. Compared with vanilla RNN, it adds various filters to selectively store or discard information in its memory, which addresses the gradient vanishing problem (Hochreiter and Schmidhuber 1997). Past work has proposed that a residual network is equivalent to simplified LSTM without gates (Srivastava et al. 2015).

2.3.3. Network Embedding (NE). NEs are used in machine learning tasks to convert discrete or categorical variables into continuous vectors (Pennington et al. 2014, Lin et al. 2015). For example, the well-known word2vec word embedding algorithm words into continuous representations (Mikolov et al. 2013). Although word embedding is used to capture similarities between words based on usage context, in our case we use NE to capture similarities between "ridership context" vectors.

2.3.4. Attention. Neural networks have a large number of input nodes. Attention mechanisms are used to focus the learner on relevant inputs to improve prediction (Vaswani et al. 2017, Dosovitskiy et al. 2021).

2.3.5. Fusion. When integrating multiple types of features (e.g., temporal, spatial) we must make a choice when to pass the features to the network for training. In early fusion, a single neural network is trained over all features, whereas in late fusion (Snoek et al. 2005), separate neural networks are trained over each feature set, and the scores are then used as inputs for another LSTM. Furthermore, because heterogeneous features are often high dimensional (Zhao et al. 2015), the framework should also be able to identify redundant and irrelevant data and select those that are more informative for the task.

2.3.6. Other Neural Networks for Time Series Modeling. In addition to RNN, other well-known deep neural network architectures that learn well from time series data directly are time-delay neural networks (TDNNs) and conditional restricted Boltzmann machines (CRBMs) (Längkvist et al. 2014). TDNNs (Waibel et al. 1989) are modified FNNs where memory of the past is implemented by extending the input with a sliding window of previous inputs. A CRBM (Taylor and Hinton 2009) is a generative probabilistic model first introduced for motion detection. Like the TDNNs' sliding window, CRBM

links a fixed size of previously visible layers to the currently hidden layer.

3. Problem Formulation and Design Principles

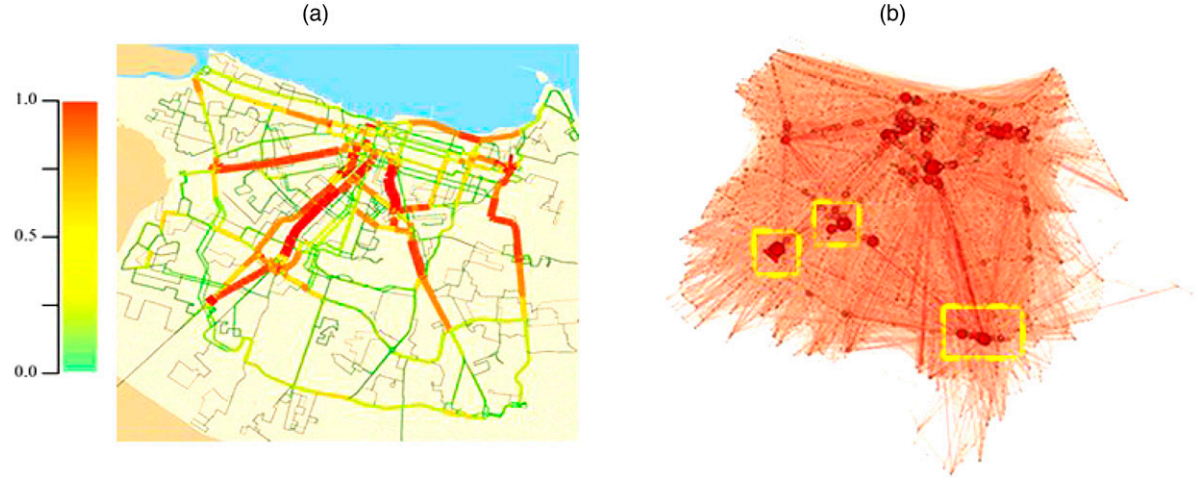
In this section, we discuss the neighbor detection method we designed to form an urban mobility graph (OD matrix). We also provide a formal definition of our research problem. Although we adopt a deep neural network design in this paper, other techniques may be adapted to solve this problem in the future. Additional details of our architecture and data set used in our evaluation are in Sections 4 and 5.

3.1. Time-Varying Mobility Graph

To capture group travel behaviors on an urban scale, we aggregated passenger flows observed during specific periods and represented them in a directed graph where each pair of vertices indicate an origin and a destination, and their weighted edges reflect the ridership. Because urban mobility exhibits strong temporal patterns, we made the graph time-varying by resetting edge weights regularly at a predefined time interval. The challenge here is how to define the vertices to form the mobility graph. One option is using all (relevant) bus stops in the transit network. This is suitable for analyzing a single route or a segment of the network. If we are interested in the harder problem of analyzing the entire network, including all stops will make the graph density extremely low, because only bus stops within a route will be directly connected. In our design, we used a directed graph to differentiate outbound/return ridership between two areas. This cannot be done if we use bus stops as vertices because typically the two bus stops in an outbound trip are different (they may be across the street, for example) with the stops in a return trip. In addition, passengers travel choices are driven by their travel purpose and bus stop accessibility. Adjacent bus stops may have similar travel attractions and accessibility. Therefore, it is more feasible to use a group of bus stops (approximating a neighborhood) to represent the origin and destination in a commute pattern. A single pair of boarding/alighting bus stops is not sufficient to capture all trips following a similar mobility demand. In the following example, we demonstrate the potential relationship between urban regions and bus stops. Figure 1(a) shows the bus stop network color-coded by normalized demand, displaying a commute pattern between southern residential neighborhoods and northern central office areas. Figure 1(b) is a mobility graph during rush hour. Vertices are bus stops arranged by geo-locations and sized by demands. The three highlighted areas in Figure 1(b) are shaped by a group of high demand bus stops. Their locations are near the main residential quarters of the city. This example shows that flows in the bus stop network can be sparse even in peak hours and adjacent bus stops share travel attractions.

Compared with using predefined urban neighborhoods, forming regions by clustering bus stops is more flexible, and can be adjusted when the transport system expands. We can also control the number of stops and routes in each region so that the distribution is balanced. When urban planners investigate a region in problem, they can quickly attribute the root cause to the right bus route and bus stops. After clustering converges, we can replace boarding and alighting stops in the trip data set with their assigned regions. To generate edges, we accumulate passenger flows in the same direction during a period of Δt

Figure 1. (Color online) Time-Varying Mobility Graph



Notes. (a) Urban bus transit network. (b) Mobility graph.

representing regular bus operation times (e.g., 5:00 a.m. to 10:00 p.m.). Vertices and edges of every $G_T^{\Delta t}$ are stored in an OD matrix, denoted as $\boldsymbol{A}_T^{\Delta t}$. The OD matrix is a key data representation that makes it possible to build the proposed model from the view of entire transit network considering both spatial and temporal correlations.

3.2. Problem of Ridership Forecasting

Our research objective is to predict short-term ridership between urban regions. Let $a_T^{\Delta t}(i,j)$ be the number of passengers traveling from origin i to destination j during the next period Δt starting at time T. The forecasted value of $a_T^{\Delta t}(i,j)$ is predicted as a function of its sequential precedents. If naïvely extended to all ODs in the city, this method will have to optimize thousands of time series simultaneously, which makes it tedious and difficult to scale for citywide forecasting. Additionally, modeling urban ridership as separate time series fails to consider spatial correlations. Fortunately, these problems are addressed in our proposed model that uses an OD matrix as a multidimensional input. Because every target value $a_T^{\Delta t}(i,j)$ is essentially a cell in the matrix $\boldsymbol{A}_T^{\Delta t}$, the model can be built from the perspective of entire bus transit network considering the spatial correlations. Thus, the research problem is changed from forecasting thousands of univariate time series to forecasting a single matrix sequence. Formally, our goal is to forecast urban-scale ridership:

$$\tilde{\boldsymbol{A}}_{T+\Delta t}^{\Delta t} = F(\boldsymbol{A}_{T-(n-1)\cdot\Delta t}^{\Delta t}, \boldsymbol{A}_{T-(n-2)\cdot\Delta t}^{\Delta t}, \ldots, \boldsymbol{A}_{T-\Delta t}^{\Delta t}), \tag{1}$$

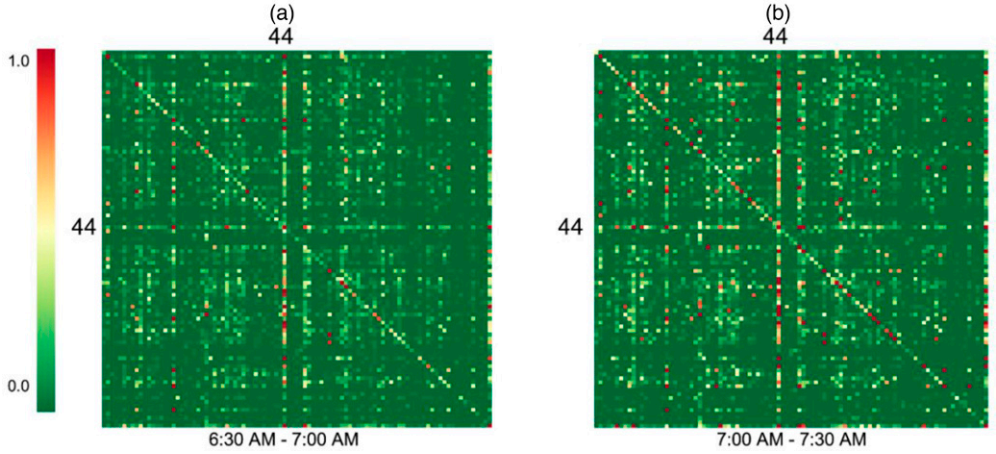
where F is a prediction model and n indicates the number of historical observations in the sequence. Please see online appendices for more details. In the remainder of this section, we review some data characteristics that pave the way for our design and provide an overview of our model framework.

3.2.1. Mass Gathering.

A mass gathering is a process where a large number of people move to a specific area during a period via different paths. In public transportation, this process can happen regularly at transfer points or rush-hour spots when most people commute. A mass gathering implies a spatial correlation between multiple origins and a particular destination. Identifying such a process can be useful for ridership forecasting in two ways. First, it indicates an on-going travel demand. The volume of inbound passenger flows to the destination could remain high for a considerable amount of time. Second, given that travel transfers are quite often on public transportation, there can be more outbound passengers at the gathering destination during the next period. In metropolitan areas such as New York, London, and Paris, 30% riders take more than one trip to arrive at their final destination when using mass transit (Guo and Wilson 2011). Although connecting trips are not explicitly labeled in our data, we found more than 30% of riders took three or more trips daily on average, implying a high transfer frequency in the city. Figure 2 gives examples of OD matrix heat maps indicating urban gathering. The OD matrices were captured at 30-minute intervals from 6:30 a.m. to 7:30 a.m. on a random weekday. In Figure 2(a), we observe many hot spots in column 44, showing that passengers were converging on this region. Then, in Figure 2(b), the convergence continues and meanwhile we see more outbound passengers leaving region 44 (more hot spots in row 44).

From a modeling perspective, a mass gathering features certain columns of an OD matrix (inbound flows to destinations) that have high value in their cells. This pattern is visually observable on the matrices' heat maps, as shown in Figure 2. However, because of the high matrix dimensionality, it is not feasible to hard-code all possible patterns for a machine to detect. To overcome this issue, we model the detection of mass gathering in an OD matrix as a visual recognition

Figure 2. (Color online) OD Matrices Indicating Urban Mass Gathering



problem and use CNNs to automatically learn the patterns. When an OD matrix is fed to a CNN, a convolutional layer will connect its kernels to part of the input and compute the dot product between kernel weights and the connected fields of the original inputs. Thus, each CNN kernel can be viewed as a gathering pattern. If the inputs match with a pattern (have high values in paired positions), the convolution result will trigger the activation neuron, implying detection of a mass gathering.

3.2.2. Temporal Patterns. Through constructing time series measured at the level of week, day and hour, previous studies find that passenger demand patterns are almost cyclical every week. Weekday and weekend patterns are similar across weeks (Xue et al. 2015). Thus, we reorganized the OD matrices to generate three multivariate time series of different intervals. A general definition of multidimensional time series is as follows:

$$TS(p,q) = \boldsymbol{A}^{\Delta t}_{T-q\cdot(p-1)\cdot\Delta t}, \boldsymbol{A}^{\Delta t}_{T-q\cdot(p-2)\cdot\Delta t}, \ldots, \boldsymbol{A}^{\Delta t}_{T-q\cdot\Delta t}, \quad (2)$$

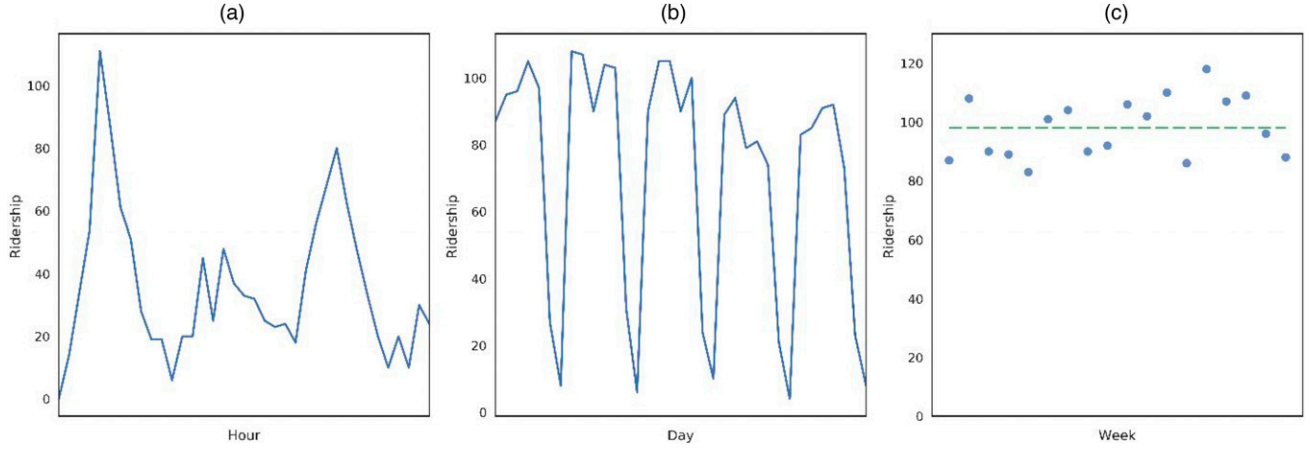
where n indicates the number of instances in the sequence and q controls the interval between steps. By adjusting the value of q, we can obtain time series at different intervals, for example, hourly, daily, weekly. For simplicity, we define hourly, daily, and weekly time series as $TS^{\Delta t}(n_h, q_h)$, $TS^{\Delta t}(n_d, q_d)$, and $TS^{\Delta t}(n_w, q_w)$, respectively.

In Figure 3, we randomly select a pair of urban regions and demonstrate how the ridership between the two regions changes over hours, days, and weeks. Figure 3(a) depicts trends of ridership at 30-minute intervals within a day. Hourly ridership in this example shows temporal dependencies on the time of day. It peaks during the morning rush hours and gains a boost around noon and during the afternoon rush hour. Figure 3(b) presents the distribution of ridership between 7:00 a.m. and 7:30 a.m. over consecutive days. The demand remains high during weekdays and drops significantly during weekends, reflecting a weekly periodicity. Figure 3(c) describes ridership from 7:00 a.m. to 7:30 a.m. on consecutive Mondays. We can see the demand distributed closely around the average (dashed line).

In the previous example, temporal dependencies exhibit differently in the three time series, and they can be more varied across thousands of ODs. Aiming to encode the time series into nonlinear high-level representations for forecasting, we used the LSTM network for its state-of-the-art performance on modeling complex multivariate sequences because of its unique gating mechanism.

3.2.3. Exogenous Impacts. Passengers travel choices can be influenced by exogenous factors include weather conditions, temperature, special events, time of day, and weekday or holiday. We define a context $\boldsymbol{e}_t$ as a group of exogenous factors at time t (e.g., rain, 30°C, football game, 8:00 p.m., Monday, not a holiday). All factors are made into nominal variables, for example, temperature is rounded to the nearest 5°C. In our deep learning architecture, contexts contribute in two phases. In the LSTM layer, contexts measure the relevance of every input in the time series with respect to the target, providing additional guidance for sequence aggregation. In the fusion layer, we pass context through an FNN to adjust the predictions from LSTM so that the final outcome takes into consideration exogenous impacts. However, the fact that exogenous factors are nominal variables makes it difficult to quantify their relevance by regular similarity measurements, hindering effective utilization of

Figure 3. (Color online) Ridership Patterns



Notes. (a) Hourly patterns. (b) Daily patterns. (c) Weekly patterns.

context. Therefore, we leveraged NE to learn real-valued vector representation of contexts, namely context embedding. The new representation makes contexts that have similar impacts on ridership (increase or decrease) close to each other in the embedding space.

Thus far, we discussed how we extract generalizable and meaningful representations to encode spatial, temporal, and exogenous factors. Taking all inputs into consideration, we derive the formal definition of predicting the OD matrix at time $T + \Delta t$ in Equation (3). The inputs are three time series of historical OD matrices and the exogeneous factors cover all time series data:

$$\tilde{A}^{\Delta t}_{T+\Delta t} = F\Big(TS^{\Delta t}(n_h, q_h), TS^{\Delta t}(n_d,\ q_d),\ TS^{\Delta t}(n_w,\ q_w),\ [e_{T-q_{w\cdot(n_w-1)\cdot\Delta t}}, \ldots,\ e_{T-\Delta t}, e_T]\Big). \tag{3}$$

Equation 3 defines the problem of single step forecasting, and we can extend it to multistep forecasting by forecasting future OD matrices one by one and feed predictions from previous steps as predictors in the next step. Hence, a multistep forecasting at time $T + k \cdot \Delta t$ is defined in Equation (4).

$$\tilde{A}^{\Delta t}_{T+\Delta t} = \begin{pmatrix} TS^{\Delta t}(n_h, q_h),\ \tilde{A}^{\Delta t}_{T+q_h\cdot\Delta t}, \ldots,\ \tilde{A}^{\Delta t}_{T+(k-q_h)\cdot\Delta t} \\ TS^{\Delta t}(n_d, q_d),\ \tilde{A}^{\Delta t}_{T+q_d\cdot\Delta t}, \ldots,\ \tilde{A}^{\Delta t}_{T+(k-q_d)\cdot\Delta t} \\ TS^{\Delta t}(n_w, q_w),\ \tilde{A}^{\Delta t}_{T+q_w\cdot\Delta t}, \ldots,\ \tilde{A}^{\Delta t}_{T+(k-q_w)\cdot\Delta t} \\ e_{T-q_{w\cdot(n_w-1)\cdot\Delta t}}, \ldots,\ e_{(k-2)\cdot\Delta t},\ e_{(k-1)\cdot\Delta t} \end{pmatrix}. \tag{4}$$

In both Equations (3) and (4), the function F is referred to the deep neural network that we designed for this problem. The neural network is composed of the aforementioned CNNs, LSTMs and NE, and these building bricks are integrated strategically to form a deeper neural network where all parameters are end-to-end trainable and optimized simultaneously.

4. Deep Learning Architecture

In this section, we introduce our deep neural network architecture for short-term ridership forecasting in more detail. Figure 4 shows the architecture of our proposed three-layer deep neural network (training phase). The first CNN component learns mass gathering patterns from the OD matrix. It contains newly designed receptive filters to infer mass gathering events automatically from each OD matrix. Intermediate results are turned into three multivariate time series, from which we learn temporal patterns in the LSTM layer. To teach LSTM how to differentiate exogenous factors, we introduce a neural attention mechanism that uses context embedding to guide the aggregation in LSTM, where context embedding is trained by considering city-level passenger demand. Last, we have a fusion layer produce a final prediction by dynamically combing the results from LSTM networks and context embedding. We compare it with a ground truth matrix and measure the loss for backpropagation training (Schmidhuber 2015).

Following the workflow in Figure 4, we discuss our customization in each neural network layer and formally describe how these neural networks process the data.

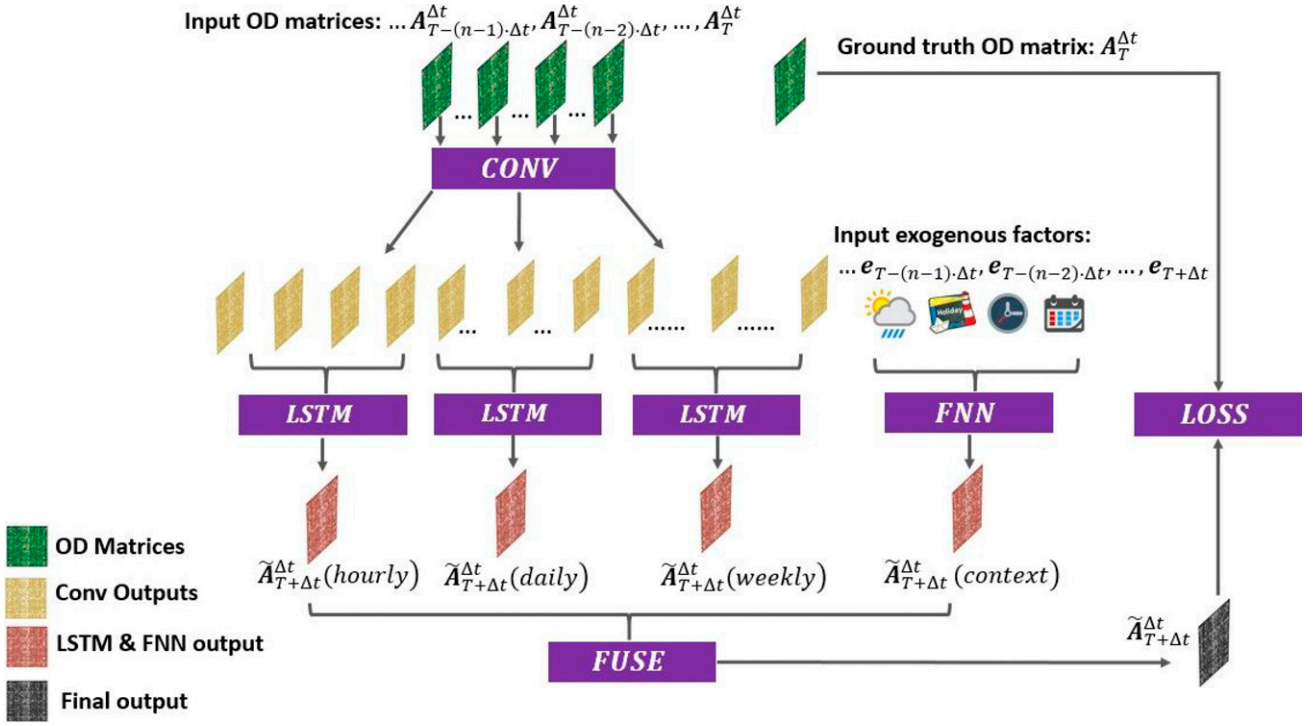
4.1. Inferring Mass Gatherings with CNN

As mentioned earlier, we learn travel transfer patterns in an OD matrix by feeding it to a CNN. In image processing problems, the shape of the kernel formally called receptive field, is usually a square of size 3×3 or 5×5, because kernels are used to exploit local correlations between nearby pixels to detect patterns such as edges and curves. Unlike natural images, cells in an OD matrix are not correlated to all surrounding cells. For example, $a^{\Delta t}_T(1,2)$ and $a^{\Delta t}_T(2,3)$ are diagonal cells in OD matrix $A^{\Delta t}_T$, measuring passenger flows from region 1 to region 2 and region 2 to region 3, respectively. The two flows are not correlated, because the two groups of passengers are disjoint. However, $a^{\Delta t}_T(1,2)$ can be correlated to $a^{\Delta t}_T(3,2)$ because the two groups of passengers both arrive at region 2 and some of them may keep traveling during next period, which would affect future ridership. In the OD matrix, cells in one column are passenger flows that share a destination. This layout inspires us to design a receptive field of size m by one where m is the number of regions. Thus, the height of a kernel is the same as the height of the OD matrix. Figure 5 depicts an example of workflow in our CNN structure. We first perform the convolution which is essentially an element-wise multiplication between the kernel weights and a full column of data in the OD matrix. Adding the kernel bias to the convolution result, we obtain the output neuron which is then connected to a rectified linear unit (ReLU) for nonlinear transformation. Formally, a ReLU output is calculated as

$$r^k_j = ReLU\left(\sum_{i=1}^{m} w^k_i \cdot a^{\Delta t}_T(i,j) + b_k\right), \tag{5}$$

where w^k_i is the i*th* weight in kernel k, b_k is a bias of the kernel, and j labels a column (region) number. Intuitively, weight values near zero imply that the corresponding ODs are less important, whereas large values indicate that the ODs are prioritized. A high value ReLU output implies a mass gathering. Therefore, the convolution operation between a kernel and a column in the

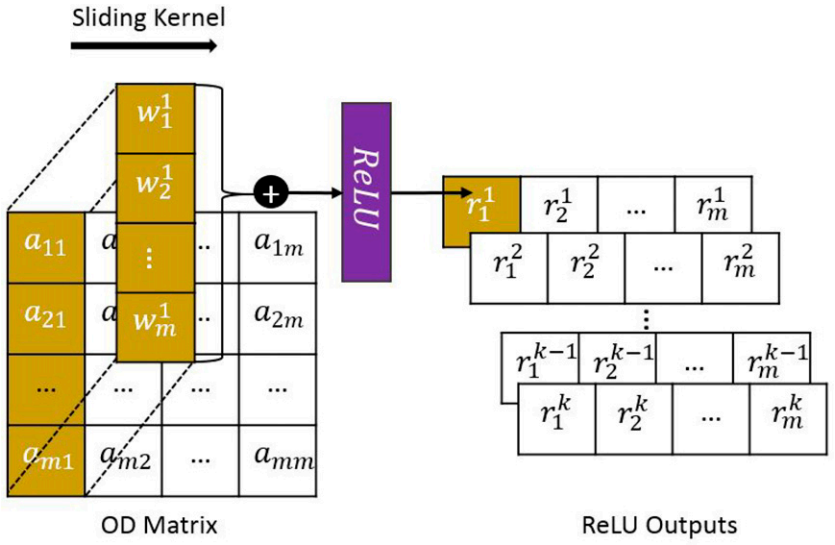
Figure 4. (Color online) Architecture of Deep Neural Network



OD matrix examines whether passenger departures from certain regions are moving toward a specific destination. After sliding across all columns in the matrix, the kernel evaluates the potential of mass gatherings at all destinations using that kernel pattern.

The proposed CNN structure does not include pooling. Each value r_j^k in the kernel feature map indicates if passengers are gathering at region j follow from origins specified by kernel k. The outputs of the CNN consist of ReLU outputs of all kernels. We concatenate CNN outputs to the flattened OD matrix to generate a new input vector, denoted as $v_T^{\Delta t} = [Vec(A_T^{\Delta t}) : Conv(A_T^{\Delta t})]$, where *Vec* is the vectorization of a matrix, and *Conv* means outputs from our CNN.

Figure 5. (Color online) CNN Structure for Encoding Transfer Patterns



4.2. Predicting Time Series in LSTM

LSTM as a special structure of RNN can selectively update or discard information in memory. This feature allows LSTM units to learn long range dependencies, for example, when two correlated inputs have a long time lag in between. In this work, we train three LSTM networks separately based on hourly, daily, and weekly time series. Without loss of generality, the following example uses hourly time series as an example (time series interval is Δt). For daily and weekly time series, the training processes are essentially the same.

The structure of a single LSTM to encode input vector $v_T^{\Delta t}$ at time T is in Figure 6(a). The purpose of an LSTM is to transform $v_T^{\Delta t}$ and previous states (factored into two parts: $C_{T-\Delta t}$ and $h_{T-\Delta t}$) into a fixed-length hidden state h_T as the output, and update cell state $C_{T-\Delta t}$ to C_T. Cell states running through the top are updated using scalar multiplication and addition operations (labeled + and × in Figure 6(a)).

The reason that an LSTM can preserve information for a relatively long time and convey information across the entire sequence is because of its "gate" mechanism that contains forget gates f_T, input gates i_T, and output gates o_T. A gate is implemented as a sigmoid function that outputs a value from zero to one to filter states. The hyperbolic tangent function (*tanh*) keeps values in states between −1 to 1. Output from previous time points (short term memory) is concatenated with the current input to update cell and hidden states as follows:

$$\begin{aligned} f_T &= sigmoid\left(W_f \cdot \left[h_{T-\Delta t},\ v_T^{\Delta t}\right] + b_f\right), \\ \tilde{C}_T &= tanh\left(W_{\tilde{C}} \cdot \left[h_{T-\Delta t},\ v_T^{\Delta t}\right] + b_{\tilde{C}}\right), \\ i_T &= sigmoid\left(W_i \cdot \left[h_{T-\Delta t},\ v_T^{\Delta t}\right] + b_i\right), \\ o_T &= sigmoid\left(W_o \cdot \left[h_{T-\Delta t},\ v_T^{\Delta t}\right] + b_o\right), \end{aligned} \tag{6}$$

where W and b in each of these functions are learnable weights and bias. When receiving a new input $v_T^{\Delta t}$ in a time series, the LSTM always concatenates it with output $h_{T-\Delta t}$ from the previous time step. The forget gate decides what portion of past information in $C_{T-\Delta t}$ can be ignored. We first obtain a candidate set $\tilde{C}_T$ for memory update via a *tanh* layer. The input gate adds into $C_{T-\Delta t}$ a new pattern $\tilde{C}_T$ learned from the current period, so $C_{T-\Delta t}$ is updated as

$$C_T = f_T \cdot C_{T-\Delta t} + i_T \cdot \tilde{C}_T. \tag{7}$$

For instance, if short-term memory and current data points suggest a demand burst pattern that conflicts with the existing cell state, the LSTM block can make the adjustment and pass it to subsequent steps by using the gate mechanism. The last part is updating the output h_T, which is also the hidden state that feeds into the LSTM (the feature of RNN). Using the output gate as filter, we calculate the new hidden state as

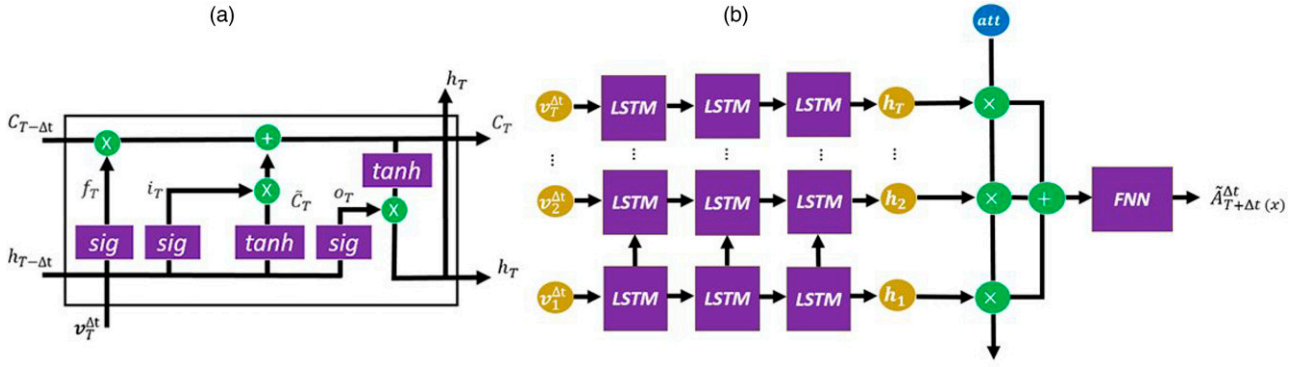
$$h_T = o_T \cdot tanh(W_C \cdot C_T + b_C), \tag{8}$$

which can be explained as selecting patterns from a set of global patterns most related to the current time.

4.3. Context Embedding Using FNN

Before we discuss the LSTM attention mechanism (next section), we share details of the FNN used generating the relevant context vectors. Measuring similarity between two contexts is a key step in our LSTM modeling. However, methods for similarity computation are not straightforward for nominal variables because different values of a nominal attribute are not inherently ordered. Conventional methods that measure similarity proportional to the number of overlap attributes fail to distinguish between the different values taken by an attribute (Boriah et al. 2008). Newer machine learning methods try to learn continuous representations of multivariate nominal concepts so that general distance

Figure 6. (Color online) LSTM Structure



Notes. (a) LSTM neurons for encoding time series. (b) Unfolded view of stacked LSTM layers with attention mechanism for ridership forecasting.

measurements can be applied (Lin et al. 2015). We designed a neural network structure to learn continuous representations for contexts. We call the new continuous representation *context embedding*.

By definition, a context is a set of multivariate nominal variables. Hence, there are a finite number of distinct contexts in our data. We define x-dimentional one-hot vectors to initialize contexts where x is the number of unique contexts. Each training instance consists of two parts, a one-hot vector of a context and the count of passengers in the transit system at time t. Our motivation is to use overall demand as the indicator to differentiate contexts.

Figure 7 illustrates the structure of FNNs for training context embedding. The hidden layer is an embedding space of a size x by z where each row represents a context embedding and z is an adjustable parameter defining the dimensions of embedding space. The output layer is single neuron that maps the embedding vector to a scalar value. During the training, each input one-hot vector serves as a lookup index that locates one row in the embedding space. Retrieved context embedding is passed to the output neuron and returns a value that is compared with the labelled data for training. After the training converges, contexts that are associated with similar passenger demand will have similar embedding representations. We measure the cosine similarity between two embeddings to facilitate the calculation of attention values. In addition, we use context embedding as an input to an FNN that maps the embedding to the same dimension as the OD matrix. Formally, the matrix is calculated as

$$\tilde{A}^{\Delta t}_{T+\Delta t}(context) = tanh(\boldsymbol{W}_{context} \cdot \boldsymbol{\lambda}_{T+\Delta t} + \boldsymbol{b}_{context}), \quad (9)$$

where $\boldsymbol{\lambda}_{T+\Delta t}$ is the embedding of context at time $T + \Delta t$. We use this matrix as a regulator to adjust predictions from LSTM networks.

4.4. Attention to Optimize LSTM Predictions Using Context Embeddings

Thus far, we explained how LSTM encodes a time series of inputs. For prediction, we use a set of vertically stacked LSTM layers to create a hierarchical feature representation such that more complex input patterns can be described at every layer. On top of the last LSTM layer is a neural attention mechanism. By default, LSTM tries to encode the entire time series, which is not necessary when inputs in time series are not equally important for prediction. As an example, suppose we are predicting ridership on a rainy Monday, then information from previous two days (Saturday and Sunday) is less relevant because of the differences in travel patterns between weekdays and weekends. Rather, the model should pay more attention to a previous rainy weekday that is more likely to have a similar travel pattern. To this end, we introduce the attention mechanism that allows LSTM to focus on certain time steps in the time series when predicting.

We use exogenous factors to calculate attention values. Let $\boldsymbol{A}^{\Delta t}_{T+\Delta t}$ be the target OD matrix to be predicted, the attention value of ith input in the time series is calculated as

$$\alpha^{n}_{T+\Delta t}(i) = \frac{exp(sim(\boldsymbol{e}_{T-(n-i)\cdot\Delta t}, \boldsymbol{e}_{T+\Delta t}))}{\sum_{j=i}^{n} exp(sim(\boldsymbol{e}_{T-(n-j)\cdot\Delta t}, \boldsymbol{e}_{T+\Delta t}))}, \quad (10)$$

where n is the length of time series, and $sim(\boldsymbol{e}_i, \boldsymbol{e}_j)$ measures the context similarity between time i and time j, which was discuss in the previous section. The *softmax* function here is used to normalize attention values in the range [0, 1] that add up to one. In general, $\alpha^{n}_{T+\Delta t}(i)$ is a weight assigned to output $\boldsymbol{h}_{T-(n-i)\cdot\Delta t}$ at time step i when LSTM aggregates outputs at each time step to generate a final output for prediction. Therefore, $\boldsymbol{h}_{final}$ for predicting $\boldsymbol{A}^{\Delta t}_{T+\Delta t}$ is calculated as

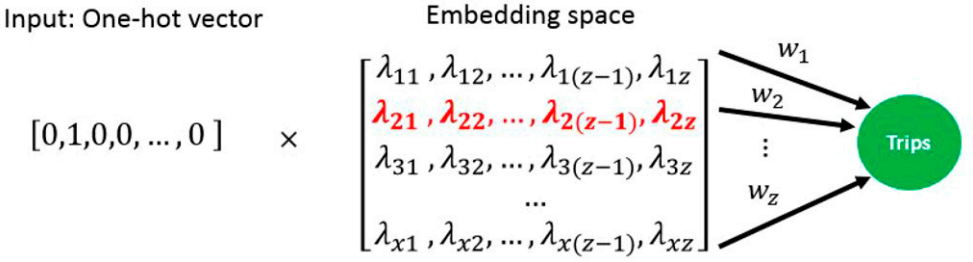
$$\boldsymbol{h}_{final} = \sum_{i=1}^{n} \alpha^{n}_{T+\Delta t}(i) \cdot \boldsymbol{h}_{T-(n-i)\cdot\Delta t}. \quad (11)$$

Once we obtained $\boldsymbol{h}_{final}$, we make a final prediction by passing $\boldsymbol{h}_{final}$ through an FNN. The predicted OD matrix is defined as

$$\tilde{A}^{\Delta t}_{T+\Delta t\,(x)} = tanh(\boldsymbol{w}_{fnn} \cdot \boldsymbol{h}_{final} + \boldsymbol{b}_{fnn}). \quad (12)$$

The workflow of the attention-based multilayer LSTM is shown in Figure 6(b).

Figure 7. (Color online) FNN for Training Context Embedding



4.5. Late Fusion Combining CNN, LSTM, FNN, and the Attention Mechanism

The last component in our prediction model is a fusion layer that combines intermediate predictions from LSTM networks and the context embedding. We prefer a late fusion method to combine results to yield a final prediction. Compared with early fusion where features from various time granularities are integrated at the feature extraction phase (and used to train a single LSTM), late fusion tends to give better performance with the price of extra learning efforts (Snoek et al. 2005). Formally, the final prediction is calculated as

$$\tilde{\boldsymbol{A}}^{\Delta t}_{T+\Delta t} = \boldsymbol{W}_h \circ \tilde{\boldsymbol{A}}^{\Delta t}_{T+\Delta t}(hourly) + \boldsymbol{W}_d \circ \tilde{\boldsymbol{A}}^{\Delta t}_{T+\Delta t}(daily) + \boldsymbol{W}_w \circ \tilde{\boldsymbol{A}}^{\Delta t}_{T+\Delta t}(weekly) + \tilde{\boldsymbol{A}}^{\Delta t}_{T+\Delta t}(context), \tag{13}$$

where $\boldsymbol{W}_h$, $\boldsymbol{W}_d$, and $\boldsymbol{W}_w$ are learnable weight matrices for hourly, daily, and weekly time series, respectively. The linear combination of Hadamard product means that the fusion layer will balance the merger of three time series at different locations while considering adjustments from context embedding as a global impact on all locations. The objective of the learning is to minimize the mean square error between prediction and ground truth defined as

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}(\boldsymbol{W}_l \circ (\tilde{\boldsymbol{A}}^{\Delta t}_{T+\Delta t} - \boldsymbol{A}^{\Delta t}_{T+\Delta t}))^2, \tag{14}$$

where $\boldsymbol{\theta}$ represents all learnable weights and biases in the model.

5. Empirical Analysis

In this section, we first introduce the data set used in the study. Then we discuss the parameter settings and experimental setup. In the experiments of ridership forecasting, we compare the performance of our proposed deep learning architecture with baseline methods. We then leverage visualization techniques to examine our neural network design. Last, we discuss how our model can potentially benefit the transport scheduling system by raising signals of overcrowdings.

5.1. Data Characteristics

We used real-world data from a large bus system of 300 routes spanning 4,800 bus stops from one of the most populated cities in Brazil. Data used were collected from approximately 2,000 buses for seven months from May through December in 2014. Most buses are equipped with GPS tracking devices sending latitude/longitude coordinates every 15–30 seconds. The AFC system supported both cash and smart card payments. The data set contains transactions at all bus stops except terminals. Citywide, passengers made 30 million trips per month on average. Deriving passenger boarding location using GPS and AFC data requires additional computation. In our data, a fare collection record indicates that passenger boarded a certain bus at approximately time t but does not indicate the bus stop where the passenger boarded the bus. To infer such information, we need to search the GPS repository for the bus location at t. This search process is nontrivial, given that a typical month of GPS data contains 200 million records. Another necessary preprocessing step is estimating the alighting location, which is not available in the raw data (the AFC system is designed to collect fares and generates a single transaction for a passenger). Here we adopt a trip-chaining method (Wang et al. 2011) that infers a prior destination from the next origin. Given that a passenger made n (multiple) trips in a day, the i*th* trip can be denoted as $< b_i, l_i >$ where b_i represents an estimated boarding stop, and l_i indicates an unknown alighting stop. If the distance between b_{i+1} to a certain stop on the route of b_i is within a walking range, we assume this stop to be l_i (the latter may represent a transfer stop on a different route or the boarding point for the return trip). Using this heuristic requires additional assumptions: passengers do not commonly take other transportation methods midtransit, and the last trip in a day marks a return trip. These assumptions are likely to hold for frequent passengers whose transit sequences contain the pattern we want to learn and given the nature of the public transport network in the city we study. In Table 1, we list some bus travel characteristics in our data. Passengers who only take one trip a day constitute less than 20% of the total cases (9% in terms of trips). Nearly 40% of passengers take the same route twice a day, implying regular commuting patterns exist in the data.

5.2. Clustering Stops into Regions

In this section we discuss the clustering algorithm to generate regions for network-level forecasting (details of our single-route forecasting approach are in Section 5.5). Before introducing the bus stop clustering algorithm, we discuss how we represent a bus stop to reflect its accessibility and travel attractions. Let $S = (s_1, s_2, \ldots, s_n)$ be a set of bus stops we want to cluster and $L = (l_1, l_2, \ldots, l_m)$ be the Point of Interest (POIs) in the city. We represent each bus stop s_i as a binary vector of m dimensions where the jth bit equals to one only if s_i and l_j are within a walk-friendly distance, which is 1 km as suggested in previous studies (Yuan et al. 2013). Hence, each binary number reflects the places that are accessible from the bus stop, implying potential travel attractions. With this representation, we measure the Jaccard distance between bus stops to perform the clustering.

Table 1. Bus Travel Characteristics

Metric	Value
Percentage of passengers only taking a single trip a day	19.42%
Percentage of passengers take the same bus route twice a day	39.80%
Quartiles of the number of trips per card per day	2, 2, 4 (trips)
Quartiles of the time difference between first and last trip	257, 521, 677

Our clustering algorithm is a modified k-means algorithm with an informed seeding strategy to avoid determining the number of clusters in advance. The seeding strategy is a procedure that iteratively updates two sets. One set $I \in S$ contains qualified bus stops that can be picked as a cluster centroid. The other set $K \in S$ contains bus stops that have already been used as centroids. K is empty when seeding initializes. Intuitively, bus stops are qualified to become a centroid if it has significant high volume (indicating a typical travel demand) and is far away from existing centroids. At the beginning of every iteration, a new I is formed by identifying qualified bus stops from Set $(S \backslash K)$. Then a qualified bus stop is randomly introduced into K. The iteration continues until I is empty. Formally, let $U(s_i)$ be the number of passengers who boarded at s_i within the interest period, and $dist(s_i, s_j)$ be the driving distance between stop s_i and s_j.

Every iteration, set I is updated as: $I = \{s_i \mid s_i \in (S \backslash K),\ U(s_i) \geq \tau_a \ and\ \forall s_j \in K : dist(s_i, s_j) \geq \tau_d\}$ where τ_a is the threshold to control the minimum boarding passengers of an initial center and τ_d controls the minimum distance between the new center and selected ones (a minimum distance ensures each cluster corresponds to a different region). A bus stop $s_i \in I$ for the next center is randomly selected based on the probability $P(s_i) \in \frac{U(s_i)}{\sum_{s_j \in I} U(s_j)}$. Repeating the seeding process until I becomes empty, we then proceed with the standard k-means algorithm. The motivation behind this seeding strategy design is to form clusters that have a high passenger base and are evenly distributed across the city. Urban planners can use their domain knowledge to adjust parameters τ_a and τ_d so that a predefined cluster number is not needed, making the parameter adjustment friendlier to domain experts compared with conventional tuning of the number of clusters.

5.3. Experimental Settings

We configure region clustering algorithm parameters τ_a and τ_d based on domain knowledge. Recall that parameter τ_d measures the minimum distances between two initial centroids. According to the city policy, bus stops should be distributed within 500 m of urban POIs. Heuristically, two bus stops that serve the same population should be a maximum of 1,000 m apart. Therefore, we set τ_d to 1,000 m to potentially separate bus stops with different travel attractions. For τ_a, we collect bus stops located in "busy" areas and use their average daily volume to set it to 600 for the experiments. Leveraging our travel attraction representations, we identify bus stops that feature business districts, schools and hospitals with a daily volume of 30. Our clustering algorithm generates 95 regions in the city. On average, each region contains 44 bus stops and three bus routes, which we consider a reasonable neighborhood size to support drill down reporting by domain experts. Table 2 summarizes the characteristics of the trips between the regions formed for this experiment.

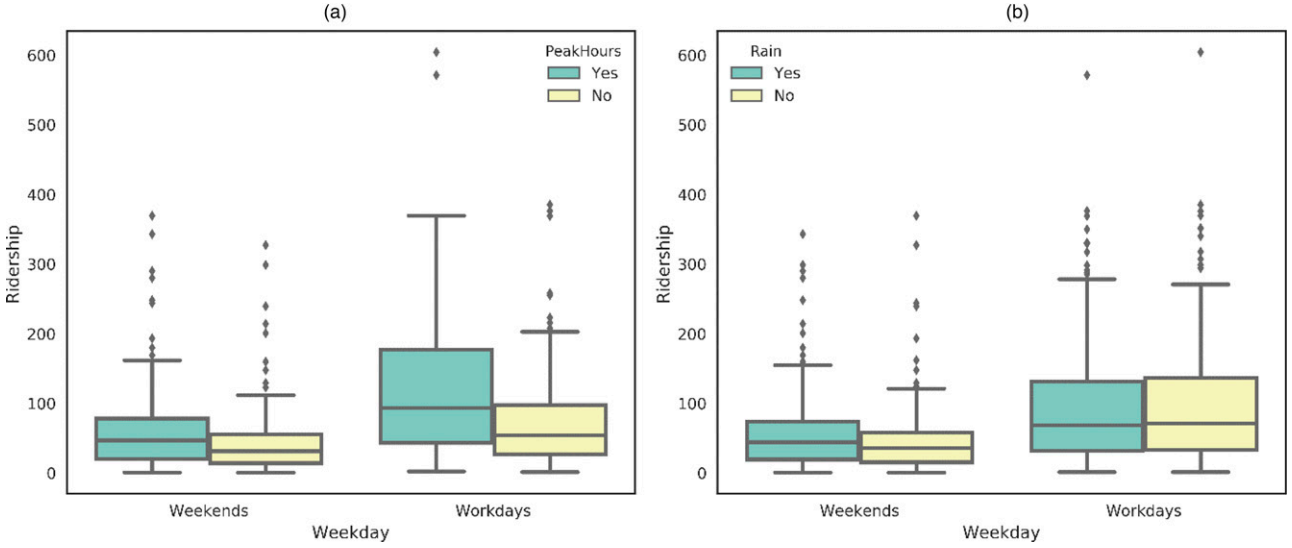
By using regions instead of individual stops, the mobility graph network density increased significantly from 0.023 to 0.665. Each mobility graph now carries denser information with less noise, which also avoids exponential growth in the input space (OD matrix dimensions). We set the update frequency of time-varying mobility graph as $\Delta t = 30$ (minutes), which is close to the average 35-minute bus journey time in the city and fits well our objective of short-term forecasting. In total, 8,200 OD matrices are derived from 200 million trips for the eight-month period from May through December. Figure 8 compares the distribution of regional outbound ridership under different circumstances, for example, rain has more impact on weekends than weekdays. In this paper, we let the machine automatically integrate exogenous impacts instead of hand-crafting patterns.

OD matrices are split to 70% training, 10% validation, and 20% testing. Although LSTM is capable of processing varying length of sequences, sequences in a minibatch usually have the same length to make training faster. In the hourly time series, we make predictions based on the last four time steps, equivalent

Table 2. OD Trip Characteristics ($\tau_a = 600$, $\tau_b = 1{,}000$)

Metric	Value
Quartiles of number of routes per OD pair	2, 4, 7
Quartiles of number of trips per OD pair per day	25, 55, 186
Percentage of trips starting and ending in the same region	10.21%

Figure 8. (Color online) Regional Ridership



Notes. (a) Distribution on peak hours. (b) Distribution on weather.

to data of the previous two hours. We use a length of six for the daily time series (i.e., data from the past week) to make a prediction. The weekly time series makes a prediction based on the last four time steps (i.e., data from the past month). This configuration yields the best performance on our testing data. Following the best practices for efficient backpropagation (LeCun et al. 1998), we normalized cell values by subtracting the mean and dividing by the standard deviation. In the evaluation phase, we calculate errors after we scale back the predicted matrices to the original values.

In principle, there is no established theory about how to choose deep neural network parameters. We use grid search and an early termination strategy to find the best setting for our experiments. The final convolutional layer has 50 kernels. The context embedding is trained using a space of 50 dimensions. Each LSTM has a total of 10,000 hidden neurons. A decaying learning rate was applied to avoid the gradient explosion problem. We terminated training when no improvement is observed on validation data for three consecutive epochs. All models converged after approximately 30 epochs with a minibatch size of 50 (about 1% of training data). To address the overfitting in training, we applied the "dropout" technique that randomly removes 20% of the hidden nodes in each minibatch training.

5.4. Results on Region-Level Ridership Forecasting

To evaluate the effectiveness of our model, we selected popular time series models as baselines: historical average (HA), SVR, SARIMA, TDNN, and CRBM. Moreover, we included three degraded deep learning models to assess the marginal effects of our neural network design. We name our model deep learning of spatiotemporal patterns (DLSP). The three degraded models were as follows: DLSP-CE (removing context embedding), DLSP-CONV (removing the CNN component), and DLSP-ATT (removing attention neurons). In the HA model, we average ridership based on contexts. For SVR, we consider eight input features based on previous literature (He 2014): ridership on the same day at time $T-\Delta t$, $T-2\Delta t$, $T-3\Delta t$; ridership on previous day at time T; ridership on same weekday at time T in the previous week; hour of day; day of week; and whether the day was a holiday or not. Both TDNN and CRBM consume hourly time series with a delay size of four, which aligns with the DLSP model.

We measure performance using the following metrics: mean absolute error (MAE), symmetric mean absolute percentage error (SMAPE), and root-mean-square error (RMSE). We prefer SMAPE over mean relative error (MRE), as zero observations are often found in our test cases. In addition to evaluating all ODs, we added comparison on subsets of ODs filtered by their variances. Our motivation here is that ODs with high variances are more difficult to predict and normally represent high demand. In other words, these represent times and places where we're more likely to see an undersupply of buses, influencing system efficiency and passenger experiences.

The results are in Table 3. We observe that our proposed deep learning model and its degraded versions outperform baselines in all aspects, excluding SMAPE, and for all ODs. Because a considerable number of

Table 3. Performance Comparison of Ridership Forecasting

	All ODs (±3.71)			Top 10% (±15.53)			Top 1% (±30.96)		
	MAE	RMSE	SMAPE	MAE	RMSE	SMAPE	MAE	RMSE	SMAPE
HA	2.093	5.999	**0.552**	10.928	17.816	0.323	26.665	42.877	0.276
SARIMA	1.993	4.306	0.644	11.775	15.128	0.386	25.379	30.432	0.374
SVR	1.891	4.137	0.639	9.369	13.099	0.345	23.033	24.512	0.309
TDNN	1.905	4.225	0.642	9.832	14.016	0.355	24.346	26.588	0.349
CRBM	1.767	3.846	0.636	8.599	11.898	0.307	20.834	23.000	0.269
DLSP	**1.299**	**2.340**	0.598	**4.241**	**6.110**	**0.172**	**8.713**	**13.289**	**0.105**
DLSP-CE	1.316	2.733	0.605	5.589	7.722	0.187	11.446	17.071	0.125
DLSP-CNN	1.431	3.096	0.622	5.867	8.152	0.191	12.872	18.462	0.131
DLSP-ATT	1.301	2.584	0.603	5.090	7.158	0.182	10.599	15.181	0.118

ODs have low ridership, for them, a small amount of absolute error can lead to a significant loss in performance (percentage error). Meanwhile, HA has relatively reliable performance on low demand and low variance pairs. However, when variance increases, advanced models improve significantly compared with HA. For example, for the top 1% ODs, DLSP obtains SMAPE of 0.105, reducing the error rate by 62% compared with HA. In comparison with other deep learning techniques, DLSP outperforms TDNN and CRBM, which shows that making effective use of data characteristics in the design process of neural networks boosts the predictive power. The results show that removing CNN, context embedding, and neural attention components increases RMSE by 38%, 28%, and 14%, respectively (RMSE at top 10% ODs). We suspect it is because the CNN component has more neurons than context embedding. Removing the CNN decreases model complexity. The relatively lower impact from removing context embedding and attention components is partly because some exogenous factors are implicitly encoded in time series, for example, hours and weekdays (in hourly and weekly time series).

For ridership forecast, we consider RMSE a better performance indicator as it gives more weight to large deviations such as outliers, which may cause false alarms of overcrowding or unanticipated travel demand bursts. Table 4 shows how much RMSE is reduced comparing DLSP against baselines. Results are all statistically significant under Wilcoxon rank-sum test.

Table 4. Wilcoxon's Test Results for the Comparison on RMSE (All ODs)

Comparison	RMSE reduction rate
DLSP vs. HA	60.99%***
DLSP vs. SARIMA	45.66%***
DLSP vs. SVR	43.44%***
DLSP vs. TDNN	44.62%***
DLSP vs. CRBM	39.16%***
DLSP vs. DLSP-CE	14.38%**
DLSP vs. DLSP-CNN	24.42%**
DLSP vs DLSP-ATT	9.44%**

**Significant at $p < 0.005$; ***significant at $p < 0.001$.

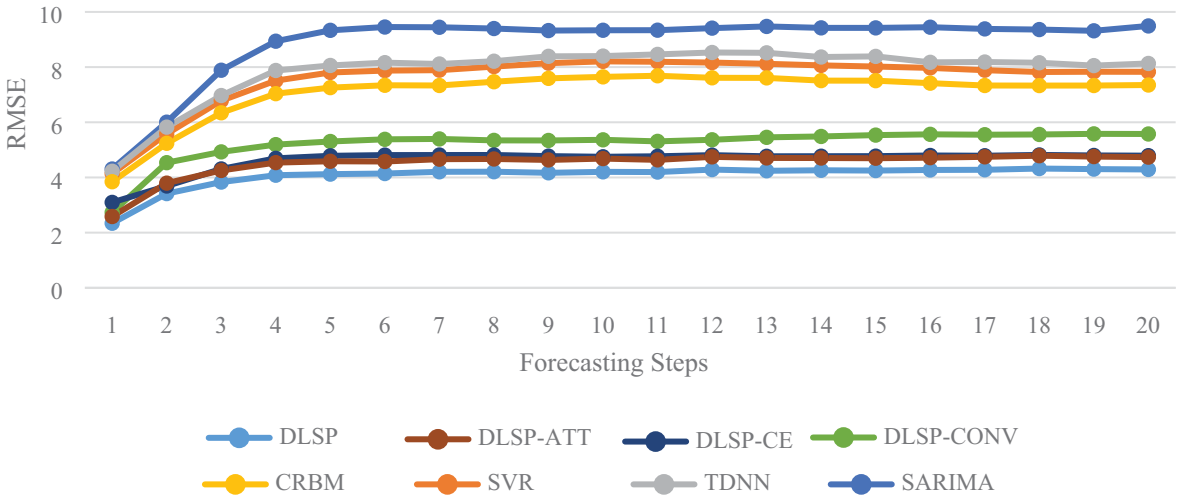
We believe multistep forecasting is preferred as it leaves enough time for urban planners to adjust transportation management based on predicted ridership. Figure 9 illustrates results of 20-step forecasting (10 hours ahead), which covers both morning and afternoon rush hours during a day. As we observe, proposed deep learning models outperform baselines consistently. Model performance decreases more in the first four steps and becomes relatively stable afterward. We notice that in the task of 10-hour forecasting, predictors in daily and weekly time series always come from ground truth. However, when the multistep prediction makes progress, the four OD matrices in hourly time series are gradually replaced with predicted ones, which increases the error. Although adding more predictors in hourly time series increases accuracy in the first four steps, the error rate keeps accumulating during the entire task, which shows the value of adding daily and weekly time series.

5.5. Single-Route OD Forecasting

The experiments in previous sections serve as a proof of concept that training heterogeneous features in an end-to-end manner is more effective in human mobility modeling compared with traditional approaches. Neighborhood ridership forecasting can help urban planners and policymakers better understand the characteristics of travel dynamics and monitor the trends of urban mobility at a macro level. In this section, we setup an experiment that applies our model on higher granular OD matrices for more actionable results.

As discussed in Section 3.2, bus stops are not ideal for constructing OD matrix on an urban scale because of the sparseness issue. This problem is alleviated when a mobility graph is built using a single route. Each vertex in the graph represents a bus stop and its counterpart stop in the opposite direction of the same route so that the graph can be fully connected. The

Figure 9. (Color online) Multistep Forecasting Evaluation



challenge of forecasting single route OD matrix lies in the fact that the data contain more random noise. First, the spatial characteristics are limited to only a small portion of the transit network, and trips connect to other routes are ignored. Second, passengers can take alternative routes for segments shared between multiple routes, making the travel patterns less stable. Conversely, ridership forecasting on single bus route can directly help adjust the bus dispatching schedules. To illustrate this further, we pick one of the highly used bus routes of 80 bus stops (both directions) and build our model based on a fully connected mobility graph of 40 nodes.

The selected route has an average of 7,000 trips per day. Quartiles of number of trips per OD pair per day are 3, 7, and 14. We trained our DLSP model using a same setting of parameters. Table 5 shows that DLSP consistently outperforms baselines. However, the error reduction rate is less compared with the results on the urban scale, which could be because of the increased randomness in the data. DLSP gained less improvements by learning from spatial and exogenous features on a small scale of transit ODs, but LSTM is still powerful in encoding multivariate time series.

This experiment provides a solution for urban planners to monitor passenger demands at certain bus stops and estimate the overcrowding to specific destinations. The information can be shared with passengers to help estimate their travel experience (potential onboard overcrowding) if they plan to take the bus. To this end, we pick one origin with the highest boarding volume and measure the MAE of ODs from it. The MAE column in Table 5 shows how our proposed model outperformed others for this task.

5.6. Evaluating Self-Taught Learning

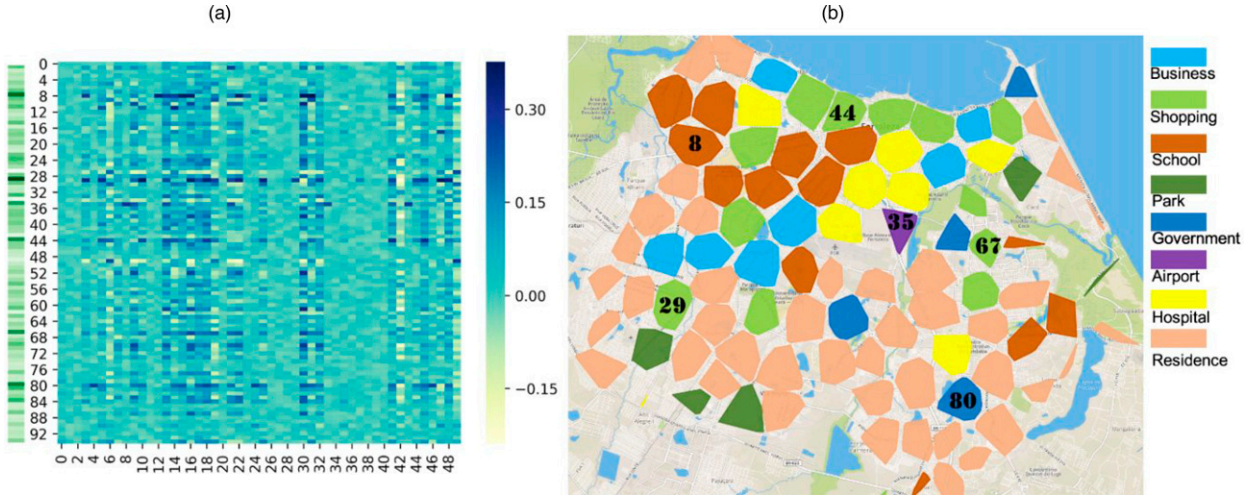
Typically, deep neural networks such as RNN are complex black box models with higher accuracy but lower interpretability than traditional models such as logistic regression (Choi et al. 2016). Indeed, interpreting high-level representations learned in neural networks is not straightforward. In this subsection, we attempt to evaluate our neural network design using visualization techniques.

Table 5. Performance on Single Bus Route ($\Delta t = 30$ minutes, $nh = 4$, $nd = 6$, $nw = 4$)

	RMSE	RMSE reduction	MAE (selected ODs)	MAE reduction
DLSP	1.641	—	1.549	—
HA	3.135	47.65%***	3.427	54.80%***
SARIMA	2.879	43.00%***	3.027	48.83%***
SVR	2.732	39.94%***	2.893	46.46%***
TDNN	2.244	26.87%***	2.219	30.19%***
CRBM	2.106	22.08%***	2.117	26.83%***
DLSP-CE	1.738	5.59%**	1.658	6.57%**
DLSP-CNN	1.779	7.74%**	1.657	6.52%**
DLSP-ATT	1.724	4.80%**	1.622	4.50%**

**Significant at $p < 0.005$; ***significant at $p < 0.001$.

Figure 10. (Color online) Visualizing Convolution Kernels



Notes. (a) Regions with large weights. (b) Urban regions highlighted by average demand.

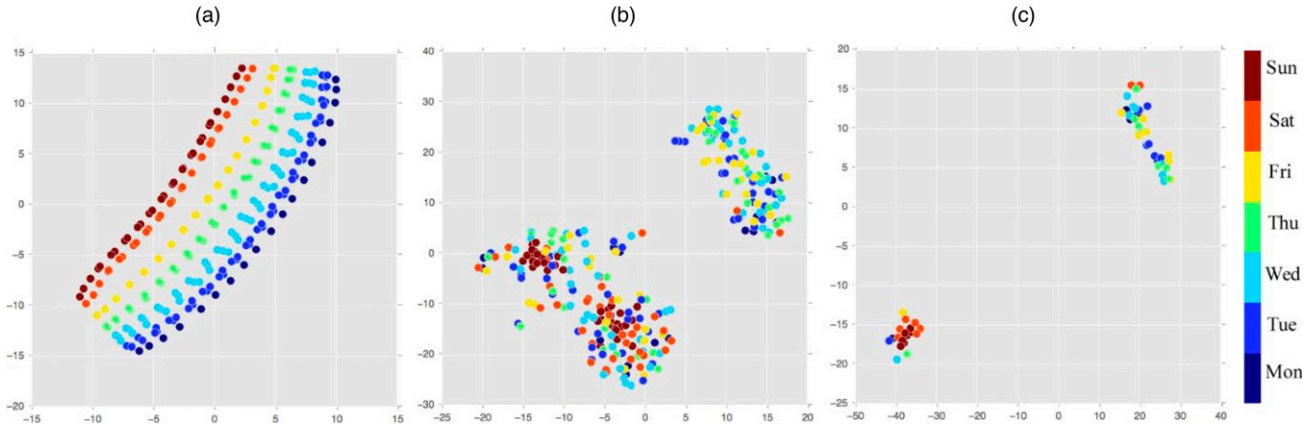
5.6.1. Convolution Kernels. Each kernel in the convolutional layer is a column vector where every element is associated with an origin. During the convolution, if large kernel weights are multiplied with ODs (cells in the OD matrix) that are experiencing high passenger volume, then the ReLU will be activated, indicating an urban mass gathering. Hence, we visualize all learned kernels in Figure 10(a) where each column represents a kernel with 95 weights. Horizontally, we can see how often a region is featured in kernels by spotting large weights in the row. The left green bar shows the average weight of all regions. Regions 8, 29, 35, 44, 67, and 80 are top ranked and highlighted in Figure 10(b) where regions are colored to show the most common POI type. We find that kernels "smartly" prioritize regions that have large passenger volumes. In particular, region 8 has a terminal station and several schools nearby. Regions 29 and 80 are surrounded by residences. These regions have a high travel demand to the central business area like region 44. Region 67 has a shopping center with a higher education institution nearby. Last, the city airport is located in region 35. Examining historical data, we found significant outbound traffic from travelers. Without explicitly feeding CNN this information, it still learns this knowledge which shows its self-taught learning efficiency. In addition, the map visualization along with POI-based region labeling gives more contextual information to large ridership flows. Urban planners can refer to this in transportation management (e.g., bus stop location, capacity adjustment).

5.6.2. Context Embedding. Contexts play a pivotal role in modeling neural attentions in LSTM. The key here is to learn their continuous representations to support similarity measurement. To demonstrate the effectiveness of context embedding, we visualize the spatial distribution of contexts in the embedding space and compare it with their raw representation. We use t-distributed stochastic neighbor embedding (t-SNE) to visualize high dimensional vectors and color contexts by weekdays. Figure 11(a) depicts how raw contexts are distributed in a two-dimensional space. After dimension reduction, time of day and day of the week are the two most discriminative features. Contexts are separated using the two features. However, their similarities cannot be accurately measured. For example, in the original representation, contexts of Friday are closer to those of Saturdays than Mondays, which does not match our prior knowledge. Conversely, we observe from Figure 11(b) that contexts in the learned embedding space are separated into two groups. Most contexts in the top-right group are weekdays. In the bottom-left group, weekend contexts are distributed closely to each other with surrounding weekday contexts. If we filter out non–rush-hour contexts, we can see a clearer separation between the weekday group and weekend group. Note that context embedding is learned from one-hot vectors that do not explicitly contain information such as day of the week and rush hour. Our model can learn effective representation to reflect these differences.

6. Conclusion

Understanding human mobility patterns is key to the implementation of intelligent transportation management and improving passenger experiences. The ability of a system to provide intelligence stems from the combination of timely data availability and the power of predictive analytics. In this paper, we proposed a

Figure 11. (Color online) Visualizing Contexts Using t-SNE



Notes. (a) Raw context. (b) Context embedding. (c) Context embedding of rush-hour contexts.

novel deep neural network architecture to learn the complex dynamics of urban mobility. The effectiveness of the proposed model is demonstrated through ridership forecasting using large-scale heterogeneous bus transportation data.

From a real-world applicability perspective, most previous work on ridership forecasting has been limited in scale because of the constraints in accessing data collection instruments such as automatic vehicle location (AVL) and automatic passenger counter (APC) devices. We leverage the power of big data analytic techniques to build our system upon commonly available digital infrastructure modules for data collection, that is, the GPS-AFC systems. This approach supported by distributed data integration techniques can produce sufficient data to train deep multilayered neural networks.

From a modeling perspective, most prior literature on urban mobility forecasting focuses on the time series ridership data to predict passenger flow. The granularity of past work is usually at the stop level, though some consider the route level or adjacent neighborhoods. This gap in knowledge is addressed in our deep neural network design. The core idea of using deep learning is to have machines learn high-level representations from spatial correlations, temporal dependencies, and exogenous factors by themselves. To the best of our knowledge, our work is a novel case of using deep neural networks for citywide regional ridership forecasting that takes advantage of an end-to-end pipeline without needing hand-crafted features.

Through representing citywide passenger flows in the time-varying mobility graph, we formulate the ridership forecasting as a matrix prediction problem. The graph representation is essential to modeling spatial correlations, as it allows using CNN to infer urban mass gatherings. Given the power of RNNs for analyzing sequential data, we design parallel multilayer LSTM networks to encode time series of OD matrices. To improve the prediction accuracy, we add an attention mechanism to make LSTM context aware. Attention is calculated based on context embedding, which contains continuous representations learned from categorical exogenous factors. The last component is a late fusion layer that merges LSTM outputs for a final prediction. We evaluate our model on real-world data collected from a large city in Brazil. Experimental results demonstrate the efficiency and superior performance of our model for region-to-region forecasting and route-level forecasting.

6.1. Implications

Our work proposes a novel artifact for microlevel predictive analytics (Abbasi et al. 2016) that can contribute to the advancement of both methodology and practice. In this paper we show how a traditional research problem on spatiotemporal data can benefit from the marriage between big data and deep learning. We develop a novel formulation of ridership patterns using time-varying OD matrices. Using such a sequential graph representation, researchers can rely on the combination of CNNs and LSTMs (or other state-of-the art prediction methods) to self-learn features from high dimensional data, which is traditionally hard to learn using feature engineering. Our system allows for the embedding of a varied mix of heterogeneous variables. Such feature embedding is highly adaptable. In a hierarchical neural network architecture, each component is customizable, for example, rewriting the loss function to optimize a domain-specific goal.

In terms of generalizability, the mobility graph is appropriate for region and route level forecasting. It also makes our model extendable to other transit modes (e.g., taxis or metros). For instance, given a broad definition of origin and destination, ridership

of multiple transit modes can be grouped together. Although the model is trained on ridership forecasting, the self-taught intrinsic representation of heterogeneous spatiotemporal data can be transferred to enable the learning of new tasks related to human mobility, for example, event detection and crowd analysis, by stacking our model with additional purpose-specific layers.

In practice, our work provides an end-to-end solution for cities seeking smart transportation management. Compared with AVL-APC systems, using GPS and AFC data can save approximately $10 million for a bus system of 3,000 vehicles. The technical requirements of our solution are modest (e.g., distributed data processing, neural network training). Considering that the relatively small Hadoop cluster in our use case was sufficient to process data promptly for an entire transportation system with 250 million records, we feel the technical prerequisites are not burdensome for a real-world solution.

6.2. Limitations and Future Work

Our study does have some limitations. In our current neural network design, the objective function did not consider the cost incurred by incorrect predictions. Future studies can be designed to incorporate more operational information to augment the objective function. Another limitation is that our study currently only considers the bus transportation system. The interactions between bus and other transit modes are not considered. It will be valuable for separate research to examine how urban mobility is impacted by other related smart city programs such as bike sharing, traffic control and road safety. For example, a system recommendation regarding dedicated bus lanes has the potential to impact motorists and bicyclists (i.e., other users of the road network).

To summarize, although there are areas for future improvement, we believe our work presents a novel deep-learning framework that can assist planners in predicting ridership and improving transportation in urban areas.

Acknowledgments

The authors gratefully acknowledge the numerous discussions and insights from the Fortaleza department of Transportation, Roberto Claudio (mayor of Fortaleza, Brazil), Haroldo Rodrigues, Paulo Goes, Ezequiel Dantas, and Luis Saboia.

References

Abbasi A, Sarker S, Chiang RHL (2016) Big data research in information systems: Toward an inclusive research agenda. *J. Assoc. Inf. Syst.* 17(2):i–xxxii.

Boriah S, Chandola V, Kumar V (2008) Similarity measures for categorical data: A comparative evaluation. *Proc. SIAM Data Mining Conf.*, 243–254.

Choi E, Bahadori MT, Schuetz A, Stewart WF, Sun J (2016) RETAIN: Interpretable predictive model in healthcare using reverse time attention mechanism. Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R, eds. *NIPS* (Barcelona, Spain). https://proceedings.neurips.cc/paper/2016/file/231141b34c82aa95e48810a9d1b33a79-Paper.pdf.

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, et al (2021) An image is worth 16x16 words: Transformers for image recognition at scale. *Proc. 9th Internat. Conf. Learning Representation.*

Du B, Peng H, Wang S, Bhuiyan MZA, Wang L, Gong Q, Liu L, et al. (2020) Deep irregular convolutional residual LSTM for urban traffic passenger flows prediction. *IEEE Trans. Intelligent Transportation Systems* 21(3):972–985.

Farkas K, Feher G, Benczur A, Sidlo C (2015) Crowdsending based public transport information service in smart cities. *IEEE Comm. Magazine* 53(8):158–165.

Gong M, Fei X, Wang Z, Qiu Y (2014) Sequential framework for short-term passenger flow prediction at bus stop. *Transportation Res. Rec.* 2417:58–66.

González MC, Hidalgo CA, Barabási AL (2008) Understanding individual human mobility patterns. *Nature* 453(7196): 779–782.

Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning* (MIT Press, Cambridge, MA).

Guo Z, Wilson NHM (2011) Assessing the cost of transfer inconvenience in public transport systems: A case study of the London Underground. *Transportation Res. Part A Policy Practice* 45(2):91–104.

Gwilliam K (2003) Urban transport in developing countries. *Transportation Rev.* 23(2):197–216.

Hasija S, Shen ZJM, Teo CP (2020) Smart city operations: Modeling challenges and opportunities. *Manufacturing Service Oper. Management* 22(1):203–213.

He W (2014) Deep neural network based load forecast. *Comput. Modeling New Tech.* 18(3):258–262.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. *Adv. Neural Inform. Processing Systems* 1–9.

Längkvist M, Karlsson L, Loutfi A (2014) A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Lett.* 42(1):11–24.

LeCun YA, Bottou L, Orr GB, Muller KR (1998) Efficient BackProp. *Neural Networks: Tricks Trade* 7700:9–48.

Li Y, Cassidy M (2007) A generalized and efficient algorithm for estimating transit route ODs from passenger counts. *Transportation Res., Part B: Methodological* 41:114–125.

Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion. *Proc. 29th AAAI Conf. Artificial Intelligence and Learning*, 2181–2187.

Liu L, Chen RC (2017) A novel passenger flow prediction model using deep learning methods. *Transportation Res., Part C Emerging Tech.* 84:74–91.

Lukoševičius M, Jaeger H (2009) Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* 3(3): 127–149.

Lv Y, Duan Y, Kang W, Li Z, Wang FY (2015) Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transp. Syst.* 16(2):865–873.

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. *Adv. Neural Inform. Processing Systems* 3111–3119.

Munizaga MA, Palma C (2012) Estimation of a disaggregate multimodal public transport Origin-Destination matrix from passive

smartcard data from Santiago, Chile. *Transportation Res., Part C Emerging Tech.* 24:9–18.

Peng TM, Hubele NF, Karady GG (1992) Advancement in the application of neural networks for short-term load forecasting. *IEEE Trans. Power Systems* 7(1):250–257.

Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. *Proc. Conf. Empirical Methods and Natural Language Processing*, 1532–1543.

Schmidhuber J (2015) Deep learning in neural networks: An overview. *Neural Networks* 61:85–117.

Snoek CGM, Worring M, Smeulders AWM (2005) Early vs. late fusion in semantic video analysis. Pereira F, Burges CJC, Bottou L, Weinberger KQ, eds. *Proc. ACM Internat. Conf. Multimed.*, Lake Tahoe, USA, 399–402.

Srivastava R, Klaus C, Schmidhuber J (2015) Training very deep networks. *Adv. Neural Inform. Processing Systems* 2377–2385.

Strover V, Edward M (2012) The impact of weather on bus ridership in Pierce County, Washington. *J. Public Transportation* 6(16):95–110.

Sun S, Zhang C, Yu G (2006) A Bayesian network approach to traffic flow forecasting. *IEEE Trans. Intelligent Transportation Systems* 7(1):124–132.

Sun L, Jin JG, Lee DH, Axhausen KW, Erath A (2014) Demand-driven timetable design for metro services. *Transportation Res., Part C Emerging Tech.* 46(September):284–299.

Taylor GW, Hinton GE (2009) Factored conditional restricted Boltzmann machines for modeling motion style. *Proc. 26th Annual Internat. Conf. on Machine Learning*, 1–8.

Ullah Z, Al-Turjman F, Mostarda L, Gagliardi R (2020) Applications of artificial intelligence and machine learning in smart cities. *Comput. Comm.* 154(February):313–323.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, et al. (2017) Attention is all you need. *Adv. Neural Inform. Processing Systems* 5998–6008.

Waibel A, Hanazawa T, Hinton GE, Shikano K, Lang KJ (1989) Phoneme recognition using time-delay neural networks. *Acoustic Speech Signal Processing IEEE Trans.* 37(3):328–339.

Wang W, Attanucci JP, Wilson NHM (2011) Bus passenger origin-destination estimation and related analyses using automated data collection systems. *J. Public Transportation* 14(4):131–150.

Wang Y, Ram S, Currim F, Dantas E, Sabóia LA (2016) A big data approach for smart transportation management on bus network. *Proc. IEEE Internat. Conf. on Smart Cities.*

Wu W, Xia Y, Jin W (2021) Predicting bus passenger flow and prioritizing influential factors using multi-source data: Scaled stacking gradient boosting decision trees. *IEEE Trans. Intelligent Transportation Systems* 22(4):2510–2523.

Xue R, Sun D, Chen S (2015) Short-term bus passenger demand prediction based on time series model and interactive multiple model approach. *Discrete Dynamics Nature Society* i:1–11.

Yang X, Zhao Z, Lu S (2016) Exploring spatial-temporal patterns of urban human mobility hotspots. *Sustainability* 8(7):674–692.

Yang X, Xue Q, Yang X, Yin H, Qu Y, Li X, Wu J (2021) A novel prediction model for the inbound passenger flow of urban rail transit. *Inform. Sci. (NY)* 566:347–363.

Yao H, Wu F, Ke J, Tang X, Jia Y, Lu S, Gong P, et al. (2018) Deep multi-view spatial-temporal network for taxi demand prediction. *Proc. AAAI Conf. on Artificial Intelligence.*

Yuan NJ, Wang Y, Zhang F, Xie X, Sun G (2013) Reconstructing individual mobility from smart card transactions: A space alignment approach. *Proc. IEEE Internat. Conf. on Data Mining*, 877–886.

Zhang J, Zheng Y, Qi D (2017) Deep spatio-temporal residual networks for citywide crowd flows prediction. Singh S, Markovitch S, eds. *AAAI* (AAAI Press, San Francisco), 1655–1661.

Zhao L, Hu Q, Wang W (2015) Heterogeneous feature selection with multi-modal deep neural networks and sparse group LASSO. *IEEE Trans. Multimedia* 17(11):1936–1948.